Colin Pham

Professor Brooks

Foundations of AI

23 September 2024

Assignment 2: Search - Question 1

Search Algorithms

Breadth First Search

- **Time Complexity:** O(|V + E|), where V represents the number of nodes on the graph and E represents the number of edges
- **Space Complexity:** O(|V + E|), where V represents the number of nodes present on the graph and E represents the edges. Though the graph can go over different nodes more than once, the fact still remains that there are V nodes and E edges actually present on the graph
- **Complete?** : Yes, since the search starts from a node and expands out until it finds the solution.
- **Optimal?** : Breadth First Search is not that optimal. There are cases where the solution is at the bottom of the graph and the queue takes a while to get to said solution.

Uniform Cost Search

- **Time Complexity:** O(m ^ (1+floor(L/e))) where m represent the number of neighbors a node has, L represents the shortest path, and e represents the shortest edge
- **Space Complexity: O(|V +** E|), where V represents the number of nodes present on the graph and E represents the edges. Though the graph can go over different nodes more

than once, the fact still remains that there are V nodes and E edges actually present on the graph

- **Complete?** : Yes, uniform cost search is complete. It will go over nodes multiple times and try multiple combinations of paths until it finds the shortest and lowest cost path possible.
- **Optimal?** : No, since the search algorithm has to try every single combination of paths, which can take a while if the space is extremely large.

Depth First Search

- **Time Complexity:** O(|V + E|), where V represents the number of nodes on the graph and E represents the number of edges
- **Space Complexity:** O(|V + E|), where V represents the number of nodes present on the graph and E represents the edges. Though the graph can go over different nodes more than once, the fact still remains that there are V nodes and E edges actually present on the graph
- **Complete?** : No, since depth first search might go down an infinite depth that gets further and further away from the goal without being able to backtrack
- **Optimal?** : Not optimal since the search is able to revisit nodes that have already been visited. This is especially true in a graph where nodes are connected by more than one connection. Also, there are cases where the solution lies in a node in a depth that is nowhere near the depths that are looked into first.

Depth Limited Search

- **Time Complexity:** O(|V + E|, level < depth limit). Basically the same as a depth first search, but the complexity is only limited to the amount of nodes and edges within the depth limit
- **Space Complexity:** O(|V + E|, level < depth limit). Basically the same as a depth first search, but the complexity is only limited to the amount of nodes and edges within the depth limit
- **Complete?** : Technically no, since limiting the depth of a search would mean that the algorithm can not try every possible combination of paths to the goal. It will find a solution, but it may not find the optimal solution
- **Optimal?** : Not optimal since the user wouldn't know what depth the best solution is located in and might prevent the AI from going deep enough to get there.

Iterative Deepening Search

- **Time Complexity:** O(|V + E|). It's basically a depth first search but with an increasing depth limit.
- **Space Complexity:** O(|V + E| ^ 2). Though the depth limit keeps increasing, the number of times the algorithm goes over nodes and edges will keep increasing the deeper the search goes.
- **Complete?** : Yes, since the search algorithm will eventually try every possible solution until the goal is reached
- **Optimal?** : It is not that optimal, especially for solutions that exist extremely deep in the graph. With how deepening search works, the time complexity will increase exponentially.

A*

- **Time Complexity:** The time complexity is a little hard to calculate since different goals and heuristics exist. How would I calculate the worst case scenario for a knowledgeable search? If I were to guess, I would say O(|V + E|) since there are some maps where a search HAS to traverse through every single node and edge to get to the optimal solution.
- **Space Complexity:** O(|V + E|). V = Nodes. E = Edges. It doesn't matter how much the search algorithm knows. If the graph is made to be traversed over entirely no matter what, then search algorithms, including A*, will traverse the entire map.
- **Complete?** I would say A* is not a complete search algorithm. Yes, it may divert a little from time to time in order to find higher peaks, but the heuristics guide it towards the goal without it straying too far. Because it is possible for it to not go through an entire graph, I will have to say that it is not complete.
- **Optimal?** : Yes, it is optimal since it